ALMA MATER STUDIORUM · UNIVERSITÀ DI BOLOGNA

FACOLTÀ DI SCIENZE MATEMATICHE, FISICHE E NATURALI
Corso di Laurea Triennale in Informatica

# Comunicazione sicura mediante il protocollo SIP; uno studio di fattibilità

Tesi di Laurea in Reti di Calcolatori

**Relatore:**
**Dott.**
**Fabio Panzieri**

**Presentata da:**
**Luca Camilletti**

**III Sessione**
**Anno Accademico 2011-2012**

*Dedicata ai miei genitori*
*che mi hanno permesso di fare*
*questa bellissima esperienza ...*

# Introduzione

Il Session Initiation Protocol (SIP) è un protocollo di segnalazione, definito dalla RFC 3261, largamente utilizzato per controllare le sessioni tra due parti (unicast) o più parti (multicast), basato su Internet Protocol (IP).

Quando SIP è nato, non è stata fatta particolare attenzione alla sicurezza, questo lo ha reso vulnerabile contro molti attacchi per anni.

Con il passare del tempo sono state proposte molte soluzioni, tanto che oggi, almeno sulla carta SIP può definirsi un protocollo sicuro. Purtroppo però molte soluzioni proposte non hanno mai trovato applicazione in uno scenario reale.

Lo scopo di questo lavoro è analizzare le misure di sicurezza adattabili a SIP, tramite uno studio del protocollo e delle sue successive estensioni di sicurezza, per poi fare uno studio di fattibilità di una possibile implementazione su di PJSIP.

PJSIP è una libreria open source che implementa uno stack SIP e uno e stack multimediale di supporto al VoIP, instant messaging e comunicazioni multimediali. La scelta di adoperare tali librerie per la mia analisi è dovuta principalmente alle sue alte prestazioni e la sua estrema portabilità; queste caratteristiche fanno di PJSIP lo strumento adatto allo sviluppo di applicazioni portabili su diverse piattaforme. Per quanto riguarda la sicurezza PJSIP spesso si appoggia a librerie esterne, come ad esempio OpenSSL e lib-

SRTP; inoltre alcune implementazione dei protocolli di sicurezza si possono adattare a PJSIP tramite pach. Purtroppo però non esiste ad oggi nessun modo per poter utilizzare il protocollo DTLS-SRTP su PJSIP. DTLS-SRTP è uno standard molto recente in grado di stabilire una connessione end-to-end senza intermediari, adatto a trasportare il traffico RTP (il protocollo maggiormente utilizzato su SIP per una sessione multimediale) garantendo un ottimo livello di sicurezza.

Durante lo studio delle estensioni di sicurezza ho rivolto particolare attenzione a DTLS-SRTP per poter analizzare la possibilità che questo venga implementato all'interno di PJSIP. Mi aspetto che l'ottima estensibilità e apertura di PJSIP si adatti senza problemi ad ospitare DTLS-SRTP

Il documento di tesi è diviso in tre capitoli di cui di seguito viene fornita una breve descrizione.

Nel primo capitolo viene fornita una descrizione dettagliata di SIP, questa parte è necessaria per capire bene come i vari strumenti di sicurezza si adattano ad esso.

Il secondo capitolo si occupa di studiare i vari strumenti applicabili a SIP per renderlo sicuro sotto il punto di vista dell'autenticazione, confidenzialità e integrità.

Nel terzo capitolo si parlerà di PJSIP e delle operazioni necessarie per poter utilizzare al di sopra di questo stak i migliori strumenti di sicurezza analizzati nel capitolo precedente. In particolare nel caso di DTLS-SRTP è stato fatto uno studio di fattibilità, descrivendo al tempo stesso i moduli e le funzioni da coinvolgere.

# Indice

# Capitolo 1

# Il protocollo SIP

In questo primo capitolo verrà analizzato il protocollo SIP, Session Initiation Protocol, in modo da poter poi comprendere con più facilità i servizi di sicurezza adottati su tale protocollo. ll protocollo SIP (Session Initiation Protocol) è un protocollo di rete basato su IP, definito dalla RFC 3261 [1], e impiegato principalmente come protocollo di segnalazione, in altre parole, per creare, modificare e chiudere sessioni multimediali di livello applicazione. Le altre applicazioni di SIP includono messaggistica istantanea, richiedere e fornire informazioni sulla presenza.

SIP è orientato allo scambio di messaggi e al Web con una struttura fortemente ispirata a HTTP che gli permette di utilizzare paradigmi di tipo "Client-Server" con l'instaurazione di comunicazioni di tipo Punto-Punto.

Uno dei suoi principali punti di forza è sicuramente la scalabilità intrinseca nella sua architettura; essa deriva dal suo sistema di instaurazione delle sessioni di comunicazione, che prevede l'utilizzo di un server per la gestione delle connessioni solo durante le fasi di instaurazione, modifica e rilascio della connessione stessa.

A connessione stabilita il flusso di dati avviene tra gli elementi terminali,

permettendo così al server di gestire numerose connessioni.

## 1.1 Soggetti

### 1.1.1 UA

Le entità partecipanti ad una sessione SIP sono gli User Agent (UA). Sono tipicamente applicazioni in esecuzione su terminali o gateway PSTN (apparati che permettono di comunicare con questo tipo di reti).

Gli UA si possono dividere in due sottogruppi: client UAC (User Agent Client) e server UAS (User Agent Server).

Uno user agent può assumere il ruolo di client o server a seconda dei casi: interpreta la parte del client nel momento in cui vuole instaurare una sessione di comunicazione con un altro user agent, assume invece il ruolo di server se invitato.

### 1.1.2 Registrar Server

È quell'entità che consente agli utenti SIP di memorizzare in un location database la loro attuale posizione; tale posizione è data dal indirizzo IP e il numero di porto associati al nome utente.

Come vedremo meglio più avanti questa operazione viene effettuata inviando un messaggio di REGISTER al Registrar Server, che una volta accettata la richiesta, si occupa di salvare i dati all'interno del database.

### 1.1.3 Proxy Server

Un SIP Proxy Server fa da intermediario tra terminale chiamante e chiamato nella fase di inizializzazione di una sessione di comunicazione, per questo

agisce sia da server (UAS) che da client (UAC).

Il proxy server riceve richieste di inizio sessione da altri UAC e si occupa di consegnarle direttamente al destinatario se conosce il terminale su cui si trova, altrimenti rigira ricorsivamente la richiesta da un altro proxy server che potrebbe conoscerla, in modo simile ad un server per l'accesso in una LAN.

I proxy server si suddividono in due categorie:

**Proxy Stateless** anche detti senza stato, inoltrano i messaggi senza conservarne una traccia, sfruttando unicamente informazioni di instradamento fornite nel messaggio stesso. I vantaggi derivati da tale approccio sono la velocità e la semplictà, ma non sono in grado di gestire le ritrasmissioni di messaggi e di realizzare forme avanzate di instradamento quali biforcazioni o attraversamenti ricorsivi.

**Proxy Stateful** considerano le informazioni ottenute dalle richieste precedenti, per questo sono più complessi e lenti, ma permettendo l'uso di alcuni servizi aggiuntivi, come la ritrasmissione e un instradamento più sofisticato.

I proxy server possono fornire anche altre funzionalità aggiuntive, come ad esempio metodi di ridirezione automatica di una chiamata in caso di mancata risposta.

### 1.1.4 Redirect Server

Il compito di un Redirect Server è quello di semplificare la localizzazione di un utente, fornendo informazioni opportune su dove si trova il destinatario ricercato.

Per ricercare gli utenti ottiene le informazioni direttamente dal location database di un registrar server e le comunica all'utente, redirezionando così la richiesta.

In figura 1.1 vengono descritte le varie operazioni effettuate.

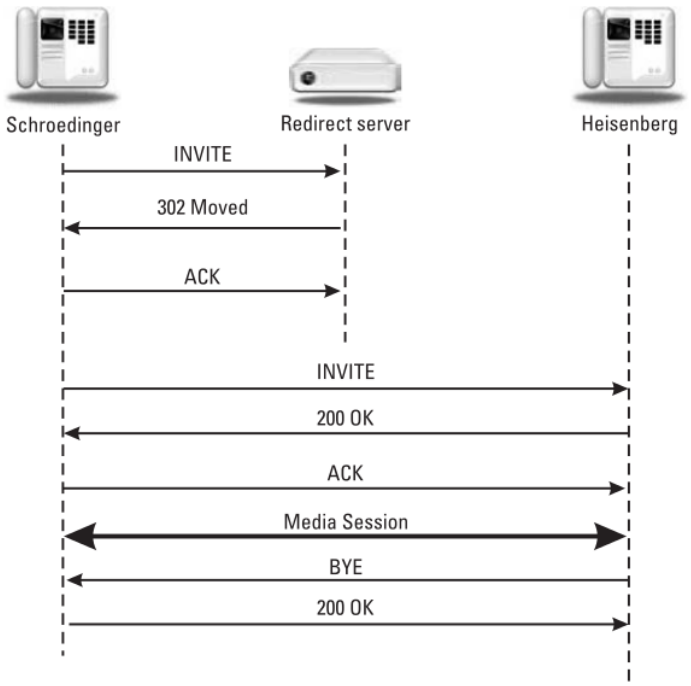


Figura 1.1: Redirect Server[26]

### 1.1.5 Location Server

Il location server risiede tipicamente sulla stessa macchina di un registrar server e contiene un database aggiornato costantemente dalle registrazioni degli utenti.

Esso viene richiamato dal redirect o dal proxy per ottenere informazioni circa le possibili locazioni dell'utente chiamato. Il locator server non utilizza protocollo sip per comunicare con il registrar ed il proxy.

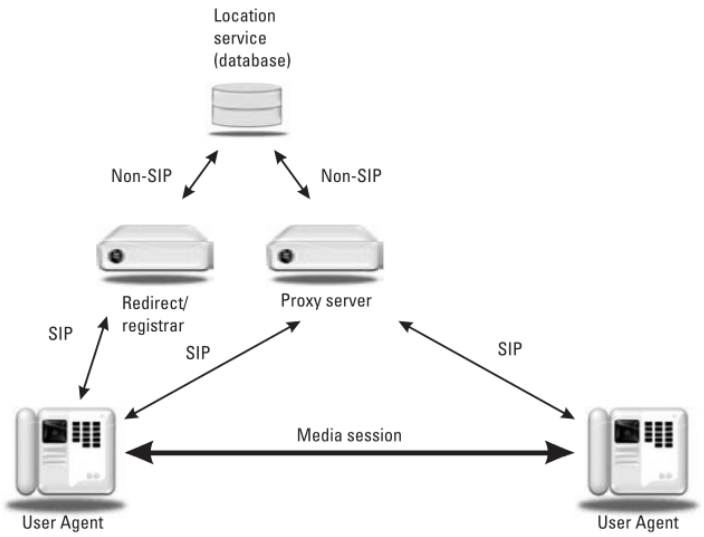


Figura 1.2: Location Server[26]

## 1.2 Indirzzamento

Un SIP-URI (SIP Uniform Resource Identifier) rappresenta lo schema di indirizzamento utilizzato per chiamare un altro soggetto attraverso il protocollo SIP.

Si presentano nella forma sip: user@host, dove user può essere un nome utente o un numero telefonico e host rappresenta un dominio o un indirizzo ip. In altre parole, un SIP-URI è il recapito telefonico SIP di un utente.

## 1.3 Struttura dei messaggi

Come detto nell'introduzione i messaggi sip hanno una struttura simile ad HTTP che verrà approfondita ne presente capitolo.

Come si può vedere nell'immagine in Figura 1.3, lo scambio di messaggi inizia con un messaggio INVITE inviato dal chiamante. Tale messaggio con-

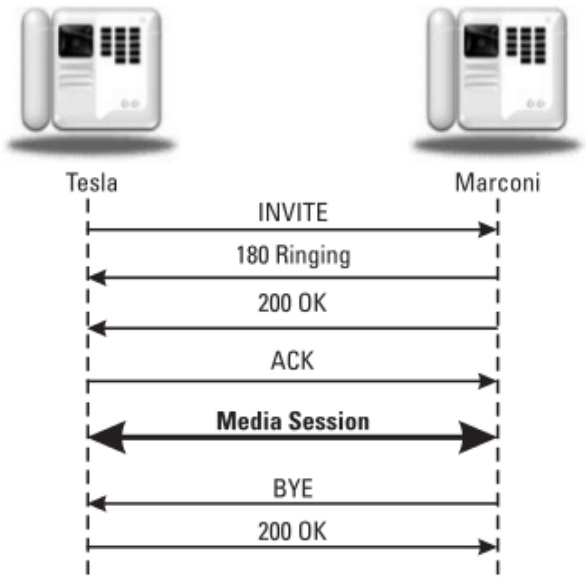


Figura 1.3: Un semplice esempio di sessione SIP[26]

tiene i dettagli sul tipo di sessione che viene richiesta rispettando la seguente struttura:

```
INVITE sip:Marconi@radio.org SIP/2.0
Via: SIP/2.0/UDP lab.high-voltage.org:5060;branch=z9hG4bKfw19b
Max-Forwards: 70
To: G. Marconi <sip:Marconi@radio.org>
From: Nikola Tesla <sip:n.tesla@high-voltage.org>;tag=76341
Call-ID: j2qu348ek2328ws
CSeq: 1 INVITE
Subject: About That Power Outage...
Contact: <sip:n.tesla@lab.high-voltage.org>
Content-Type: application/sdp
Content-Length: 158
```

```
v=0
o=Tesla 2890844526 2890844526 IN IP4 lab.high-voltage.org
s=Phone Call
c=IN IP4 100.101.102.103
t=0 0
m=audio 49170 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

La prima riga viene chiamata start line, in questo caso indica che il messaggio è un invite e che l'utente da contattare è Marconi@radio.org; SIP/2.0 è la versione del protocollo usata dal terminale che ha generato il messaggio.

Il campo Via, serve a tenere traccia del percorso compiuto dalla richiesta.

Il campo successiovo è Max-Forwards; Consiste semplicemente in un intero, che viene decrementato ad ogni passaggio su di un server SIP che riceve e inoltra la richesta; il suo scopo è quello di individuare eventuali loop.

From e To identificano il chiamante e il chiamato.

Call-ID è, un identificativo del dialog, permette cioè di individuare i messaggi appartenenti alla stessa chiamata. Cseq, definisce un ordine tra i messaggi, e utile per la gestione delle ritrasmissioni. Content-Type e Content-Length definiscono rispettivamente il contenuto del messaggio SIP (codificato tipicamente in SDP[2]) e la dimensione in byte di tale carico utile.

Il campo Contact è opzionale ed è utile al chiamante per indicare al chiamato la sua posizione corrente su chi desidera essere contattato.

Dopo la linea bianca di separazione troviamo il message-body codificato in SDP, che consente la negoziazione dei parametri della sessione.

### 1.3.1 Tipo di richeste

INVITE è un esempio di messaggio di richiesta SIP. Ci sono sei richieste definite nell'RFC 3261 (INVITE compresa) e altre definite in esenzioni di tale RFC.

I metodi di richiesta fondamentali, definiti nella RFC 3261, sono:

**INVITE** come abbiamo visto serve per richiedere l'apertura di una sessione dove all'interno di tale richiesta sono presenti informazioni relative alla sessione che si vuole stabilire, definite tramite il protocollo Session Description Protocol.

**ACK** conferma la ricezione di una risposta definitiva relativa ad una precedente richiesta di INVITE, è il quarto passaggio in figura 1.3.

Nel momento in cui il destinatario riceve un INVITE, viene generata immediatamente una risposta provvisoria (nel esempio in figura 1.3 è 180 Ringing), mentre l'eventuale risposta definitiva (200 OK) viene generata dal destinatario solo quando l'utente ha effettivamente accettato la chiamata. A quel punto viene generato un ACK per confermare la sua presenza e la ricezione del messaggio.

Il corpo dell'ACK può contenere una descrizione della sessione nella usuale codifica SDP. In caso contrario, il chiamato deve considerare ancora validi i parametri di sessione presenti nell'INVITE ricevuto precedentemente.

**OPTIONS** richiede le capacità di uno UA. Un client si può servire di questo metodo per ottenere informazioni relative allo stato di un utente e alle funzionalità da esso supportate.

**BYE** richiede la terminazione di una sessione attiva; nell'immagine (figura 1.3) è il penultimo passaggio che avviene subito dopo la sessione media e può essere inviato indistintamente dal chiamante o dal chiamato.

**CANCEL** Viene utilizzato quando uno UA decide di interrompere un tentativo di chiamata. Una richiesta CANCEL non ha effetto nel caso in cui la sessione sia già stata instaurata.

**REGISTER** Questo messaggio viene inviato da un client ad un Registrar Server per fornire informazioni sulla sua attuale posizione.

Le seguenti richieste non fanno parte dell'RFC 3261, bensì di alcune sue estensioni:

**INFO[3]** questo metodo viene usato per scambiare informazioni senza modificare lo stato della sessione. Le informazioni di interesse possono essere inserite in un apposito campo di intestazione INFO o semplicemente nel corpo del messaggio.

**PRACK [4]** consente ad un client di riscontrare la ricezione delle risposte provvisorie, che possono essere così trasmesse in maniera affidabile, al pari di quelle definitive.

**SUBSCRIBE e NOTIFY [5]** SUBSCRIBE richiede la sottoscrizione ad un evento al fine di ricevere notifiche attraverso richieste di tipo NOTIFY. La sottoscrizione implica la creazione di una sessione. Il messaggio contiene anche un campo che specifica la validità temporale della sottoscrizione, che può essere rinnovata all'interno della stessa sessione.

**UPDATE [6]** consente ad un client di aggiornare i parametri di una sessione senza modificare lo stato della sessione.

**MESSAGE** [7] consente ad un utente di inviare messaggi istantanei nell'ambito di una sessione già stabilita. Il contenuto dei messaggi viene inviato all'interno del corpo della richiesta utilizzando il formato MIME (Multipurpose Internet Mail Extensions).

**REFER** [8] con questo messaggio un client può invitare il destinatario a contattare un altro indirizzo, indicato in un apposito campo di intestazione denominato Refer-To. Questo meccanismo consente a SIP di supportare una serie di funzionalità, come il trasferimento di chiamata.

### 1.3.2 Risposte

Ogni richiesta SIP deve ricevere una risposta, fatta eccezione per le richieste di tipo ACK, che rappresentano le conferme. Ecco un tipico esempio di risposta SIP:

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP 167.180.112.24:5060
From: sip:alice@wonderland.com
To: sip:bob@domain.com
Call-ID: a2e3a@wonderland.com
CSeq: 34 INVITE
Content-Type: application/sdp
Content-Length: 0
```

Come si vede, il messaggio è molto simile ad una richiesta se non per la startline, che presenta, oltre alla versione del protocollo, un codice di stato e una sua descrizione testuale. È importante notare che nella risposta i campi From e To non sono invertiti, come si potrebbe essere indotti a pensare. Ciò consente ai server SIP di mettere la risposta in relazione con la richiesta,

mentre la funzione di distinguere i due messaggi è affidata alla prima riga dell'intestazione.

I codici di stato sono degli interi compresi tra 100 e 699 e identificano le risposte inviate da un server a un client. Essi si dividono in sei categorie, a seconda del contenuto informativo:

**INFORMATIONAL (1XX)** Si tratta di risposte provvisorie inviate ad un client per informarlo che una richiesta è stata ricevuta ma la relativa elaborazione non è ancora terminata e quindi non è disponibile al momento una risposta definitiva. In seguito alla ricezione di una riposta provvisoria, che viene generata oltre un tempo di attesa di 200 ms, il client è tenuto ad astenersi da ritrasmissioni della richiesta e ad attendere ulteriori informazioni sull'esito dell'elaborazione.

**SUCCESSFUL (2XX)** La richiesta è stata accolta con successo. La risposta definitiva inviata dal server è 200 OK.

**REDIRECTION (3XX)** Questa classe di risposte fornisce informazioni riguardanti la nuova posizione dell'utente o i servizi alternativi che potrebbero portare a buon fine la chiamata.

**REQUEST FAILURE (4XX)** Si tratta di precise risposte di fallimento fornite da un determinato server. Il client non dovrebbe riformulare la richiesta senza modifiche (e.g. Aggiungendo le autorizzazioni del caso). In particolare tramite il codice 401 Unauthorized si indica al client che la richiesta necessita di un'autorizzazione da parte di un UAS o di un registrar server.

**SERVER FAILURE (5XX)** Il fallimento della richiesta è dovuto ad un errore del server.

**GLOBAL FAILURE (6XX)** Il server ha informazioni definitive riguardo all'utente desiderato ma non quelle relative alla particolare richiesta effettuata.

# Capitolo 2

# Sicurezza

Per garantire una comunicazione sicura sopra il protocollo SIP, esso deve integrare alcuni servizi di sicurezza relativi principalmente ai requisiti di autenticazione, riservatezza e integrità.

L'IETF non ha integrato alcun meccanismo di protezione specifico nel protocollo, tuttavia nell'RFC 3261 suggerisce l'impiego di alcune soluzioni standard facilmente adattabili al caso di SIP.

Successivamente con l'RFC 3329 [9] il protocollo è stato esteso con nuove funzionalità per la negoziazione dei meccanismi di sicurezza.

In questo capitolo analizzo alcuni servizi di sicurezza per SIP, standard e non (DTLS [10]), nelle fasi principali di una comunicazione SIP. Prima di questo spiego in maniera molto sintetica il significato di alcuni termini che sono usati.

**Firma digitale** [11] La firma digitale rappresenta un sistema di collegamento tra un autore e documenti digitali tale da garantire il cosiddetto non ripudio e al contempo l'integrità del documento stesso.

**Message digest (impronta digitale o fingerprint)[11]** Per impronta dig-

itale di un documento digitale si intende il risultato di una funzione hash (pubblica) su quel documento.

**Certification Authority (CA)[11]** Una Certification Authority (CA), è un ente di terza parte, abilitato a rilasciare un certificato digitale utilizzando la crittografia asimmetrica, in cui una delle due chiavi viene resa pubblica all'interno del certificato (chiave pubblica), mentre la seconda, univocamente correlata con la prima, rimane segreta e associata al titolare (chiave privata). Una coppia di chiavi può essere attribuita ad un solo titolare.

**Certificato digitale[11]** Un certificato digitale è un documento elettronico che attesta, con una firma digitale, l'associazione tra una chiave pubblica e l'identità di un soggetto (una persona, una società, un computer, etc). La firma digitale viene apposta da un'Autorità Certificativa (CA).

## 2.1 Modello di sicurezza sip

La sicurezza in SIP inizia attraverso l'autenticazione. L'autenticazione, è il processo tramite il quale un soggetto verifica l'identità di un altro partecipante che vuole comunicare attraverso una connessione. Generalmente avviene prima dell'instaurazione di un canale ma, per garantire una piena sicurezza, vi è anche una fase di autenticazione che avviene quando già il canale è instaurato e le parti sono già in fase di scambio di dati.

In SIP ci sono differenti strade per autenticare un UA, oltre a quelle analizzate di seguito, un messaggio SIP può considerarsi autentificato anche nel caso in cui viene ricevuto attraverso un tunnel IPSec o VPN.

### 2.1.1 SIP Digest Autentication

SIP Digest Autentication [12] si basa sullo standard per l'autenticazione del protocollo HTTP. Un UAS può chiedere ad un client di inviare nuovamente la richiesta, questa volta però dimostrando di essere a conoscenza di un segreto condiviso. Questo segreto non compare mai all'interno di un messaggio SIP, il client dimostra di esserne a conoscenza inviando un hash MD5 calcolato su alcuni dati. Il calcolo avviene in modo da fornire l'autenticazione o anche l'integrità (in questo caso l'MD5 comprenderà anche i dati inviati), a seconda dei requisiti di sicurezza imposti dal UAS, ma dipende in ogni caso il segreto. È possibile vedere un esempio in figura 2.1.

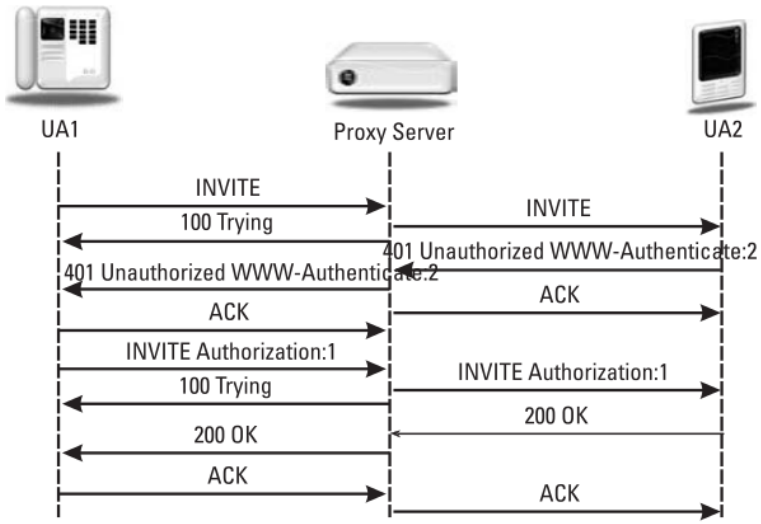


Figura 2.1: Digest Autentication[26]

L'INVITE iniziale riceve una risposta di tipo 401 (Unauthorized) contenente un header Authenticate. L'UAC invia un ACK per completare la transazione e rinvia l'INVITE con le sue credenziali nel campo Autenticate.

Il Digest Autentication può essere utilizzato anche in fase di REGISTER per prevenire un attacco di tipo hijacking. Infatti su sip un utente può effettuare una registrazione anche per conto di una terza parte.

Un malintenzionato potrebbe allora chiedere ad un registrar server di associare l'indirizzo SIP della vittima ad un proprio device IP, eventualmente dopo aver annullato una precedente registrazione, compromettendo così la riservatezza dei dati e la disponibilità del servizio: le successive richieste dirette alla vittima arriveranno infatti all'attaccante, che avrà accesso alle informazioni altrui, privandone al contempo il legittimo proprietario.

### 2.1.2 TLS

Tra i vari meccanismi di protezione suggeriti dall'IETF è presente anche TLS (Transport Layer Security) che può essere utilizzato per garantire autenticazione, riservatezza e integrità nello stesso modo in cui viene utilizzato da HTTP per la navigazione web sicura. TLS può essere visto come due protocolli separati:

TLS handshake, utilizzato per configurare la connessione, eseguire l'autenticazione e generare un segreto condiviso;

l'altro protocollo gestisce il di trasporto e la criptazione dei dati attraverso il segreto condiviso.

Il protocollo di handshake utilizza certificati digitali per garantire l'autenticazione, mentre il protocollo di trasporto usa i cifrari AES o 3DES per i dati.

Ora vediamo come viene utilizzato in una sessione sip per poter autenticare un client: Un client dopo aver aperto una connessione TLS sulla porta 5061 può richiedere al server un certificato. Se il certificato è firmato da un CA riconosciuta dal client, esso viene usato per autenticare la chiave pub-

blica del server. Il server a sua volta può richiedere un certificato al client. Ad ogni modo, a meno che il client non sia un altro proxy server, è poco probabile che sia in possesso di un certificato. Come risultato, tipicamente TLS viene usato per autenticare solo una parte. Per avere autenticazione reciproca è possibile integrare un meccanismo di Digest Autentication.

In aggiunta TLS fornisce anche protezione di integrità su base hop-by-hop a host connessi logicamente tramite protocolli di trasporto connection-oriented. Poiché in SIP i meccanismi di trasporto sono specificati hop-by-hop, un terminale che invia una richiesta su un canale TLS non ha la certezza che lo stesso protocollo di sicurezza sia utilizzato su tutte le connessioni logiche che lo collegano, attraverso i vari proxy, al destinatario. A questo inconveniente si può ovviare ricorrendo ad indirizzi SIP del tipo SIPS URI, della forma sips: user@host, che consentono di richiedere l'uso di TLS lungo l'intero percorso di un messaggio dal mittente al destinatario.

Purtroppo però Secure SIP (SIPS) ha avuto poco successo sia per il mancato supporto di TLS su molti apparati SIP sia per il ritardo dovuto all'impiego di TCP come protocollo di trasporto unito all'infrastruttura PKI. DTLS è il suo equivalente su trasporto datagram, di conseguenza può garantire lo stesso livello di sicurezza sia durante il trasporto sia durante la fase di handshake, esso non è ancora uno standard per SIP, ma potrebbe diventarlo in futuro.

### 2.1.3 S/MIME

S/MIME [14] è uno standard nato per la protezione delle e-mail basato sulla firma digitale e su tecniche di crittografia a chiave simmetrica e asimmetrica, che consente di soddisfare i requisiti di autenticazione, riservatezza e integrità. Nella definizione del protocollo SIP [1], S/MIME è il meccan-

ismo raccomandato per l'autenticazione e la cifratura dei messaggi SIP. Un user agent può utilizzare S/MIME per autenticarsi allegando all'intestazione un'entità S/MIME del tipo multipart/signed, la quale contiene a sua volta il body MIME in chiaro e una entità S/MIME del tipo application/pkcs7-signature, che costituisce la firma digitale. In questo modo è garantita oltre all'autenticazione anche l'integrità, ma non la riservatezza. Per ottenere anche tale requisito si allega allo stesso modo un'entità S/MIME (sempre multipart/signed), che contiene però, al posto del messagge body in chiaro, la sua versione cifrata, ottenuta usando il tipo application/pkcs7-mime col parametro smime-type settato a enveloped data. La seconda parte dell'allegato avrà invece la stessa forma vista prima (firma digitale del corpo MIME). Questa tecnica ha un inconveniente: un'entità SIP che non supporta S/MIME non sarà in grado di leggere il contenuto del messaggio.

Il problema maggiore di S/MIME è dovuto alla disponibilità di un infrastruttura sicura per la certificazione e la distribuzione delle chiavi pubbliche, tanto da causarne una mancata implementazione nonostante sia in circolazione da molti anni.

### 2.1.4 Enhancements SIP

L'Enhancements for Authenticated Identity Management [13] definisce un meccanismo per identificare in modo sicuro l'origine di una richiesta SIP, senza la necessità di instaurare preventivamente alcun tipo di associazione. Lo scopo è quello di consentire al ricevente di una chiamata di autenticarne il mittente, anche per le comunicazioni inter-dominio e senza l'onere di creare un certificato per ciascun utente. In pratica si basa sull'uso dei certificati solo da parte dei Server: ciascun Proxy autentica autonomamente le richieste dei propri utenti (ad esempio tramite il classico HTTP Digest), poi calcola l'hash

su alcuni campi di intestazione (tra cui il From) firmandolo con il certificato del dominio, e lo inserisce in un apposito Header denominato Identity. Così facendo in pratica autorizza il mittente del messaggio ad utilizzare l'indirizzo presente nel campo From. Il Proxy si occupa inoltre di inserire in un nuovo campo Identity-Info le informazioni necessarie per acquisire il proprio certificato del dominio, ad esempio:

```
Identity-Info: <https://proxy1.dominio.it/proxy1.cer>;alg=rsa-sha1
```

A questo punto il Proxy di destinazione (o il Client) è in grado di verificare la firma contenuta nel campo Identity della richiesta SIP ricevuta.

## 2.2 Certificate Service

Alcune delle soluzioni appena indicate (TLS, DTLS, S/MIME) possono offrire una buona sicurezza se si potesse aggirare in qualche modo le problematiche nella gestione dei certificati. Queste problematiche includono:

1. come rilasciare certificati di per gli UAS senza intercorrere in costi tipici dei certificati commerciali. È ragionevole avere un certificato CA-signed in pochi proxy e server di dominio, ma richiederli ad ogni UA comporta un costo proibitivo.

2. come installare e gestire certificati all'interno degli endpoint.

3. visto che non è consigliato avere un certificato CA-signed è necessario un meccanismo per convalidare quelli che saranno auto-prodotti.

4. un utente con più dispositivi SIP necessita di un meccanismo di sincronizzazione delle chiavi private. E nel caso in cui una nuova chiave venga emessa, deve essere distribuita in tempo reale a tutti i dispositivi dell'utente.

Un tentativo per risolvere questo problema è affrontato nel RFC 6072 (Certificate Management Service for SIP)[15]. Questo approccio utilizza SIP per archiviare, recuperare, e convalidare i certificati, trovando cosi una soluzione per tutti i problemi precedentemente elencati.

1. gli UAS producono dei certificati self-signed, che non comporta nessun costo per l'utente. Tuttavia a questo punto è necessario rafforzare l'identità su SIP attraverso l'Enhanced SIP in cui solo proxy e server di dominio sono in posseso di certificati CA-signed.

2. i certificati vengono recuperati attraverso SIP grazie all'uso di SUSCRIBE. Il certificato viene ricevuto dal server SIP nel corpo di un messaggio di un NOTIFY. Inoltre la sottoscrizione permette eventuali modifiche o aggiornamenti del certificato in tempo reale attraverso NOTFY aggiuntivi. Quando viene utilizzato per memorizzare le chiavi private, esse possono essere criptate attraverso una pass phrase in modo che sia solo l'utente a poter utilizzare la chiave anche nel caso in cui il certificato del server è stato compromesso.

3. il certificato auto-firmato non può essere convalidato dal certificato stesso. Il NOTIFY consegnato dal proxy server ha un campo identity che può essere utilizzato per validarlo.

4. Tutti i dispositivi utilizzati dall'utente possono iscriversi (attraverso SUSCRIBE) al certificate server. Se vengono generate nuove chiavi è necessario avvertire il server attraverso un PUBLISH, che poi si occupa di inviare NOTIFY agli altri dispositivi. Come risultato tutti i dispositivi sono sincronizzati sullo stesso certificato.

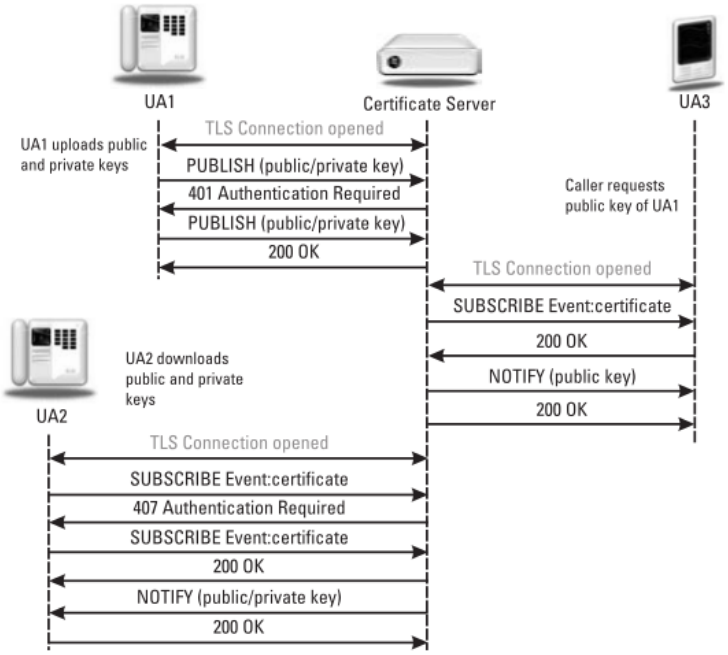


Figura 2.2: Certificate Service[26]

L'immagine (figura 2.2) mostra un esempio dove UA1 e UA2 sono associati allo stesso utente (utilizzano la stessa chiave pubblica e privata), mentre UA3 è un altro utente che ha accesso alla sola chiave pubblica.

## 2.3 Trasporto

Nel paragrafo precedente ci siamo occupati di analizzare l'autenticazione, riservatezze e integrità delle segnalazioni; la capacita di stabilire sessioni multimediali sicure è un problema correlato e altrettanto importante, di cui si parla in questa sezione.

I due protocolli di sicurezza più comuni per IP sono IPSec e TLS. È possibile utilizzare SIP per stabilire entrambi questi protocolli, anche se solo TLS è stato completamente standardizzato dal IETF. Come stabilire una

sessione attraverso il protocollo TLS è definito in nel RFC 4572 [16] ed è un'estensione della strategia utilizzato per stabilire una connessione TCP. Un esempio di offerta SDP è fornita qui di seguito:

```
m=image 54111 TCP/TLS t38
c=IN IP4 192.0.2.2
a=setup:passive
a=connection:new
a=fingerprint:SHA-1 4A:AD:B9:B1:3F:82:18:3B:54:02:12:DF:3E:5D:49
:6B:19:E5:7C:AB
```

L'attributo a=setup viene utilizzato per definire il ruolo di chi ha inviato il messaggio mentre a=connection:new indica che deve essere instaurata una nuova connessione TCP invece di riutilizzarne una già esistente. a=fingerprint contiene un fingerprint del certificato che verrà utilizzato per l'autenticazione. Questo campo è progettato per utilizzare certificati autofirmati. È importante notare che questo fingerprint a valore solo nel caso in cui la sessione è autenticata e offre un meccanismo di protezione per l'integrità. In pratica o deve essere firmato con S/MIME o deve essere in uso l'Enhanced SIP Identity.

### SRTP

RTP [17] è il protocollo maggiormente utilizzato da SIP per trasportare il traffico multimediale. Esso fornisce funzioni di trasporto di rete end-to-end adatte per le applicazioni di trasmissione di dati in tempo reale, come ad esempio: audio, video o dati di simulazione. RTP, per quanto effettui un ottima pacchettizzazione di flussi audio e video, non garantisce nessun tipo di integrità e sicurezza dei dati, per questo l'IETF ha standardizzato un protocollo specifico, il protocollo in questione si chiama SRTP[18].

Il Protocollo SRTP è progettato per adattarsi specificatamente alle applicazioni real-time e offre confidenzialità, integrità, autenticazione e protezione replay per il traffico RTP ed RTCP. SRTP usa chiavi simmetriche, che devono essere negoziate durante lo scambio offer/answer in SIP.

L'algoritmo utilizzato per la crittografia è AES in counter-mode (AES-CTR) utilizzando chiavi a 128 o 256 bit. Questo tipo di algoritmo permette che la crittografia venga eseguita in parallelo con l'elaborazione dei codec-media, riducendo cosi la latenza. Il problema maggiore di SRTP è dovuto all'impossibilita di adoperare un proprio algoritmo di keyagreement. Per questo che si rende necessario l'utilizzo e l'integrazione di un protocollo aggiuntivo, ad esempio: SDES, MIKEY, DTLS-SRTP e ZRTP, il cui compito è la negoziazione dei parametri di sicurezza in un contesto crittografico.

### DTLS

DTLS [10] è un protocollo adatto per rendere sicure le comunicazioni al di sopra di un trasporto datagram garantendo una sicurezza pari a TLS. Nonostante DTLS non viene preso in considerazione come trasporto per una sessione multimediale in SIP, esso si adatta meglio rispetto a TLS al tipo di dati che tipicamente vengono trasportati. Infatti la maggior parte delle applicazioni che usano SIP come ad esempio videoconferenza, telefonia via internet e giochi on-line, sono sensibili al ritardo. DTLS presenta vantaggi anche nei confronti di IPsec. IPsec infatti è difficile da includere nelle applicazioni dal momento che viene eseguito in kernel space. Al contrario DTLS è implementato in user space e questo garantisce una maggiore portabilità e facilita di installazione. DTLS può instaurare una connessione cifrata point-to-point (tenendo in considerazione il problema dei certificati) adatta a trasportare in maniera sicura il traffico multimediale RTP, senza che sia

necessario un meccanismo di scambio delle chiavi come su SRTP, questo a scapito di un maggiore overhead.

## 2.4 Scambio chiavi SRTP

SRTP definisce come i pacchetti RTP devono essere cifrati e protetti contro le intercettazione, ma non specifica nel dettaglio come deve essere scambiata e concordata una chiave di sicurezza comune tra i due utenti. In questo paragrafo verranno descritti i sistemi di scambio chiavi più diffusi e sicuri.

### 2.4.1 SRTP/SDES

SRTP/SDES standardizzato con l'RFC 4568 [19] è il sistema di scambio chiavi più diffuso . SDES è un protocollo di scambio chiavi molto semplice, in cui uno dei due utenti propone all'altro una chiave per la cifratura SRTP sul canale si segnalazione SIP. In seguito all'accettazione da parte dell'altro utente, inizia il vero e proprio processo di cifratura del flusso dati. Ovviamente la connessione di segnalazione SIP deve essere protetta attraverso cifratura e autenticazione delle connessioni per evitare che terzi possano carpire la chiave di comunicazione scambiata. Questo è il motivo per cui lo scambio di chiavi SDES avviene esclusivamente attraverso canali di comunicazione protetti in SIP/TLS con, autenticazione del server attraverso un certificato digitale, esattamente come accade quando si utilizza l'HTTPS.

Nonostante lo scambio chiavi avviene attraverso una comunicazione protetta SDES rimane un protocollo poco sicuro. Infatti SIP/TLS garantisce integrità e riservatezza solo su base hop-by-hop, questo vuol dire che è suf-

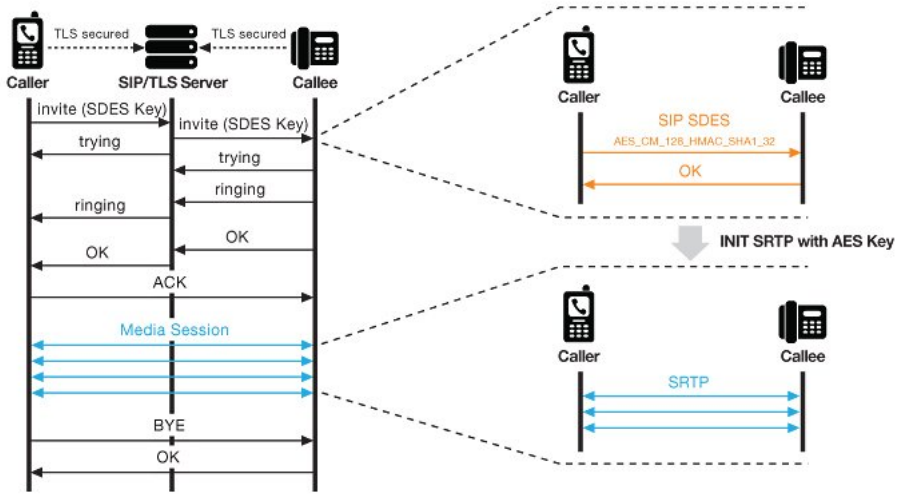


Figura 2.3: SRTP/SDES[25]

ficiente che la sicurezza sia compromessa su un solo host nel percorso tra sorgente e destinazione per compromettere la riservatezza.

### 2.4.2 DTLS-SRTP

Come già anticipato DTLS può gestire completamente la sicurezza di una sessione multimediale, ma essendo un protocollo più generico rispetto a SRTP è meno ottimizzato per l'uso specifico con RTP; questa soluzione (DTLS-SRTP) può vantare un minore overhead rispetto all'uso esclusivo di DTLS. DTLS-SRTP [20] può essere visto sia come un nuovo modo di gestione delle chiavi per SRTP sia come un nuovo formato dei dati RTP specifico per DTLS. Ogni sessione DTLS-SRTP contiene una connessione DTLS e uno o due flussi SRTP, a seconda se il flusso dati è monodirezzionale o bidirezionale. Un sessione DTLS-SRTP protegge soltanto i dati trasportati al di sopra di una singola connessione UDP (una sola coppia di porte sorgente-destinazione),

ad esempio se viene usato RTP Control Protocol (RTCP) che solitamente viene spedito su una porta diversa rispetto a RTP, sono necessarie due sessioni DTLS-SRTP ( RFC 5711 definisce il multiplexing del traffico di RTP e RTCP in una singola porta UDP, in questo caso è possibile utilizzare una sola sessione DTLS-SRTP). Lo schema generale di DTLS-SRTP è il seguente: per ogni RTP o RTCP i peer creano una connessione DTLS attraverso il DTLS handshake. Le chiavi ottenute dall'handshake DTLS vengono inserite nello stack SRTP. Una volta che la connessione DTLS è stata stabilita i pacchetti sono protetti e lo scambio dati può iniziare. Tra una singola coppia di partecipanti è possibile che ci siano diverse sessioni dati (in una video chiamata viene inviato sia un flusso audio che un flusso video) e per come è strutturato il protocollo sono necessarie più sessioni DTLS-SRTP. Visto che le operazioni di crittografia a chiave pubblica per l'handsakes DTLS hanno grandi costi computazionali bisognerebbe ridurle al minimo. In questo caso specifico è possibile ammortizzare questi costi semplicemente avviando un solo handshake e stabilire se successive connessioni DTLS riprendendo i dati della prima.

Il protocollo SIP in questo caso non viene più utilizzato per lo scambio chiavi come invece avviene su SRTP/SDES, ma soltanto per stabilire i partecipanti alla comunicazione. DTLS e di conseguenza DTLS-SRTP presenta il problema della distribuzione dei certificati che però può essere aggirato seguendo l'RFC 5763.

### Basi per stabilire una connessione SRTP utilizzando DTLS

L'obbiettivo del rfc 5763[21] è quello di fornire una tecnica di trattativa delle chiavi che permette una comunicazione criptata tra due devices tra cui non c'è mai stata una precedente relazione senza la necessita della

distribuzione di certificati firmati da una ben nota CA a tutti i dispositivi.

La sessione multimediale viene trasportata al di sopra di una connessione DTLS che è stata autenticata reciprocamente, quindi entrambe le parti devono essere in possesso di certificati. Questi certificati vengono utilizzati esclusivamente per validare le publik key dei peers. Tutto questo è necessario perché DTLS non ha un metodo diverso per trasportare le chiavi pubbliche. Per aggirare il problema di procurarsi un certificato da una CA, essi possono essere auto-generati. Una volta che si è in possesso di un certificato, bisogna inviare il fingerprint attraverso SIP. L'impronta digitale del certificato inviata attraverso SIP viene successivamente confrontata con il certificato presentato nell'DTLS handshake cosi da poter verificare che sia la stesso. Tuttavia per evitare un attacco MiTM è necessario una qualche forma di protezione che protegge l'integrità dei dati trasportati al di sopra di SIP. SIP Identity come descritto nell'RFC 4474 o S/MIME forniscono il massimo livello di sicurezza perché non sono suscettibili alle modifiche di terze parti.

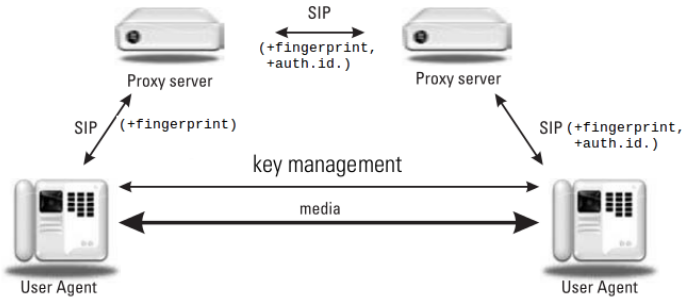


Figura 2.4: dtls-srtp[23]

Vediamo un esempio pratico. Alice invia un proposta SDP sopra SIP a Bob. Alice che utilizza un certificato auto-firmato per la comunicazione includerà l'impronta digitale di tale certificato nel corpo dell'SDP e lo invia su

un canale che garantisce un meccanismo di identificazione e integrità. Quando Bob riceve l'offerta i peer stabiliscono un certo numero di connessioni DTLS (a seconda del numero di sessioni multimediali) con reciproca autenticazione (cioè i certificati devono essere esibiti da entrambi i lati). Una volta che la connessione DTLS è stata stabilita Bob può verificare le credenziali di Alice confrontandole con l'impronta digitale ottenuta tramite SDP; se il confronto ha esito positivo Bob invia un messaggio di risposta SDP a Alice e può iniziare con la trasmissione dati. Una volta ricevuto il messaggio anche Alice confronta l'impronta digitale con le chiavi DTLS e in caso positivo anche lei inizi la trasmissione.

### 2.4.3 Zrtp

ZRTP [22] come fa anche DTLS-SRTP è un protocollo che non si basa su uno strumento di segnalazione come SIP per lo scambio chiavi, ma esse vengono scambiate direttamente tra i peer attraverso l'algoritmo Diffie-Hellman. A SIP viene lasciato solamente il ruolo di stabilire chi siano i partecipanti. Il vantaggio di scambiare le chiavi direttamente tra i peer permette di escludere completamente terze parti, compresi gli operatori di rete, garantendo una maggior sicurezza di cui gli unici responsabili sono esclusivamente i partecipanti alla comunicazione . A differenza di DTLS-SRTP però esso utilizza un protocollo di key-agreement che non necessita di un infrastruttura PKI o autorità di certificazione per eseguire uno scambio di chiavi in maniera sicura. L'algoritmo Diffie-Hellman permette di rilevare un attacco MiTM, attraverso la visualizzazione di una short authentication string (SAS), che gli utenti possono verificare confrontandola verbalmente durante la chiamata. Inoltre possiede la proprietà di perfect forward secrecy, distruggendo le chiavi di cifratura al termine di ogni sessione, precludendo in questa maniera

la compromissione retroattiva del traffico telefonico, anche qualora la chiave venga scoperta in futuro.

# Capitolo 3

# Analisi di implementazione su PJSIP

## 3.1 PJSIP

PJSIP [23] è una libreria open source che implementa uno stack SIP e uno stack multimediale di supporto al VoIP, instant messaging e comunicazioni multimediali. La scelta di adoperare tali librerie per la mia analisi è dovuta principalmente alle sue caratteristiche. PJSIP infatti assicura altissime prestazioni, unito ad un ottima portabilità (funziona su architetture a 32 bit, 64bit, big endian e little endian) e dimensioni contenute. Questa caratteristiche fanno di PJSIP un candidato adatto ad essere installato su un gran numero di dispositivi, anche con processori poco potenti, come ad esempio Smartphone e Tablet, permettendo di scrivere una sola volta l'applicazione che usa SIP. Inoltre PJSIP è rilasciato sotto licenza GPL 2.0 ed è dotato di una documentazione esaustiva, con tanto di spiegazioni articolate sull'architettura.

### 3.1.1 Struttra PJ

Spesso anche in questa tesi si fa riferimento a PJSIP riferendosi a tutta la suite PJ. Essa oltre ad offrire uno stack SIP (per l'appunto PJSIP) offre anche uno stack multimediale e altre librerie che possono essere utili allo sviluppo di applicazioni, in questo capitolo verranno descritte le principali librerie che compongono PJ.

PJSIP: rappresenta uno stack SIP che supporta un insieme di features ed estensioni del protocollo SIP.

PJLIB: rappresentante la libreria a cui le restanti si appoggiano. Si occupa di garantire funzionalità di base (es. L'astrazione rispetto al sistema operativo sottostante). Può essere considerata una replica della libreria libc con l'aggiunta di alcune features come la gestione dei socket, funzionalità di logging, gestione thread, mutua esclusione, semafori, critical section, funzioni di timing, gestione eccezzioni e definizione di strutture dati di base (liste, stringhe, tabelle, ecc...).

PJLIB-UTIL: anch'essa come PJLIB è una libreria di appoggio, ma a differenza della prima implementa funzioni più complesse utili per la crittografia come gli algoritmi: SHA1, MD5, HMAC, CRC32. Ed anche funzioni per il parsing e la manipolazione di testi.

PJMEDIA: questa libreria si occupa del trasferimento e della gestione dei dati multimediali. Tra le sue caratteristiche principali vi è la gestione degli stack RTP/RTCP (supporta SRTP grazie a libSRTP).

PJSUA: rappresenta il livello più alto fungendo da wrapper verso le altre librerie. Inoltre agevola notevolmente la scrittura di applicazioni.

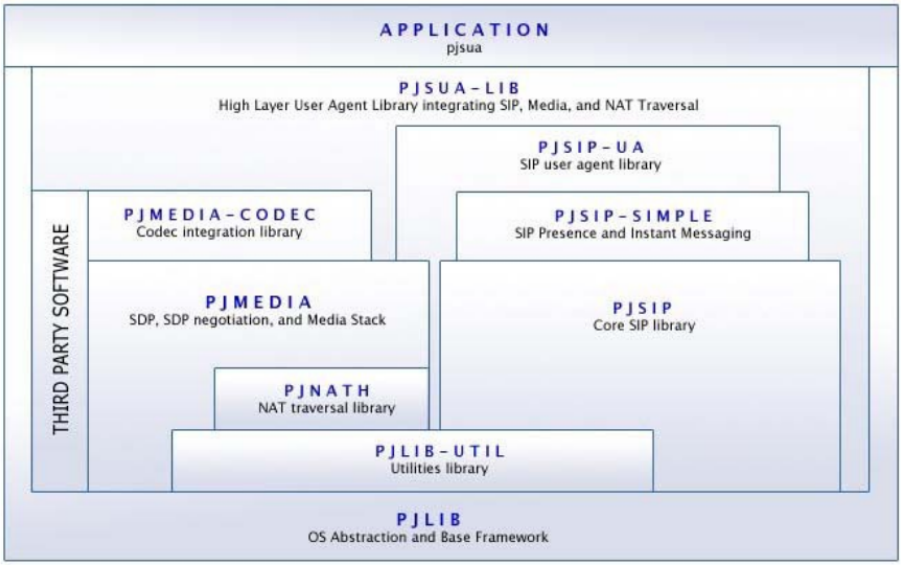


Figura 3.1: Stack delle librerie PJ[23]

### 3.1.2 Architettura di PJSIP

In questo paragrafo viene spiegata in maniera più approfondita l'architettura di PJSIP. Nonostante PJSIP sia sviluppato in C che non è un linguaggio orientato agli oggetti, la libreria PJSIP è altamente modulare e strutturata a livelli.

#### Endpoint

Al centro della libreria c'è l'endpoint (figura 3.2) che si occupa di gestire varie mansioni:

- gestisce la memoria, si occupa cioè di allocare la memoria per i vari moduli sip,
- implementa al suo interno un temporizzatore, con cui schedula eventi che notificherà ai relativi moduli SIP interessati.

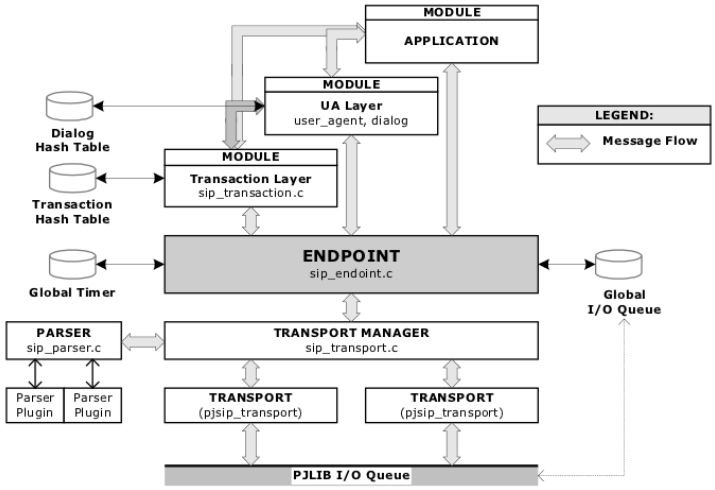


Figura 3.2: Collaboration Diagram PJSIP[23]

- gestisce le varie istanze dei moduli di trasporto e controlla il parsing dei messaggi e la stampa.

- gestisce i moduli PJSIP, che sono la struttura primaria attraverso cui estendere la libreria per fornire nuove funzionalità.

- riceve messaggi dal transport manager e li ridistribuisce attraverso i moduli che implementano i livelli dello stack SIP

### Transport

Il framework transport viene utilizzato per inviare e ricevere messaggi attraverso la rete. Esso può essere esteso per utilizzare nuovi tipi di trasporto. Teoricamente ogni tipo di trasporto può essere aggiunto al framework anche non limitato a TCP/IP. Il fulcro del livello di transport è il Transport Manager che si occupa di svolgere vari compiti:

- gestisce direttamente la vita dei transport basandosi su un sistema di conteggio delle reference ed un timer di inattività.
- gestisce la creazione di tutti i transport,
- gestisce i pacchetti provenienti dai vari transport per poi passarli all'endpoint,
- si occupa di trovare il transport adeguato per l'invio dei messaggi basandosi sul tipo di trasporto e l'indirizzo remoto
- crea nuovi tipi di trasporto dinamicamente a seconda delle esigenze.

La libreria pjsip alloca un solo transport manager per endpoint. Il transport manager normalmente non risulta visibile alle applicazioni che devono utilizzare le funzioni esposte dall'endpoint. Le altre entità presenti in questo livello sono i vari transport come ad esempio TCP o UDP e il transport factory che viene utilizzato per creare il collegamento dinamico con l'endpoint remoto.

### Transection

Transection gestisce le fasi di transazione in base ai messaggi SIP in entrata e in uscita settando delle funzioni di callback che sono necessarie al monitoraggio delle transazioni.

### UA

In questo livello è presente una classe user_agent e una classe dialog. Il suo compito principale è quello di creare le sessioni necessarie ad implementare correttamente le fasi di una comunicazione SIP.

### I/O Queue

I/O Queue non appartiene direttamente a PJSIP, ma comunica direttamente con lo strato Transport. Il suo compito principale è quello di fornire un un insieme di API per gestire le operazioni di I/O attraverso socket.

### Moduli

I moduli sono il mezzo con cui vengono distribuiti i messaggi SIP tra i vari componenti software. Tutti i componenti in PJSIP compreso lo strato transection e dialog sono implementati come moduli. Il meccanismo usato per la distribuzione dei messaggi è semplice ma al contempo potente: i messaggi in ingresso vengono distribuiti a tutti i moduli, a partire da quello con la priorità più alta, finché ogni modulo non finisce di processare il messaggio. I messaggi in uscita vengono distribuiti sempre seguendo l'ordine di priorità, prima che questi vengano trasmessi sull'interfaccia di rete. In questo modo ognuno può aggiungere delle modifiche al messaggio, oppure registrarlo per scopi di logging.

## 3.2 SRTP SU PJSIP

Come abbiamo visto SRTP definisce in quale modo i pacchetti RTP devono essere cifrati e protetti contro le intercettazioni, ma non specifica nel dettaglio come deve essere scambiata e concordata una chiave di sicurezza comune tra i due utenti. L'unico mezzo per scambiare le chiavi su PJSIP senza applicare alcuna patch o modifica è SDES, RFC 4568. Questo oltre a rende necessario lo scambio esclusivamente attraverso canali di comunicazione SIP/TLS con i relativi problemi che questo comporta non garantisce la piena confidenzialità.

Le osservazioni fatte nel capitolo precedente ci mostrano metodi migliori e più efficaci per lo scambio delle chiavi. Nei paragrafi successivi verrà analizzata la possibilità di implementarli sullo stack pjsip.

### 3.2.1 DTLS-SRTP SU PJSIP

Questo paragrafo si occupa, come esplicitato dal titolo, di esaminare la fattibilità di un implementazione DTLS-SRTP su PJSIP, verranno inoltre descritti i moduli e le funzioni da coinvolgere.

PJSIP è uno stack SIP, e come ci si può aspettare non offre un grande supporto alla crittografia. L'unica libreria che implementa funzioni utili alla crittografia è PJLIB-UTIL, tuttavia propone un ristretto numero di algoritmi come ad esempio: SHA1, MD5, HMAC, CRC32. Per concedere supporto a protocolli come SRTP e TLS si appoggia a librerie esterne, nello specifico a OpenSSL e libSRTP. DTLS-SRTP è stato specificamente progettato in modo che questo possa essere implementato con la semplice unione di uno stack DTLS e uno stack SRTP e l'apporto di minime modifiche. Dunque nel caso di PJSIP l'handshake viene effettuato grazie all'aiuto di OpenSSL, successivamente verranno estratte le chiavi ottenute per poi passarle a libSRTP. Le modifiche fondamentali per la negoziazione DTLS-SRTP sono state aggiunte su OpenSSL dalla versione 1.0.1 (che al 19-01-2012 è arrivata alla versione Beta 2). Per quanto riguarda la libreria libSRTP non sono necessarie modifiche; in una recente versione, sono state aggiunte delle funzioni di test per verificarne il corretto comportamento con DTLS-SRTP.

La trasmissione dei messaggi in PJSIP, avviene attraverso moduli denominati Transport, ognuno dei quali si occupa della gestione di un determinato protocollo, come ad esempio UDP (transport_udp.c) e TCP (transport_tcp.c).

Pjsip non offre la possibilità di utilizzare un trasporto di tipo DTLS, ma è possibile abilitare tale funzionalità imitando sip_transport_tls.c che si occupa del trasporto su TLS. Questo è realizzabile solo quando TLS è offerto dalla libreria OpenSSL (in Symbian si appoggia a CsecureSocket), che ha aggiunto il supporto a DTLS dalla versione 0.9.8. sip_transport_dtls.c sarà del tutto simile alla sua controparte TLS, ma il metodo SSL da usare in OpenSSL è in questo caso DTLSv1_method (o successivi quando saranno implementati). Per quanto riguarda SRTP è implementato all'interno di PJMEDIA; quest'ultimo mette a disposizione diverse funzioni per adoperare SRTP direttamente, anche al di fuori di una sessione SIP. PJSIP deve essere ulteriormente esteso con un nuovo tipo di trasporto, sip_transport_dtls-srtp.c che si deve occupare di effettuare il DTLS handshake (attraverso il nuovo sip_transport_dtls.c) e successivamente inserire le chiavi cosi ottenute nello stack SRTP come descritto nel RFC 5764. In alternativa è possibile creare un solo transport aggiuntivo che all'interno gestisca sia il socket DTLS che quello SRTP.

### 3.2.2 Stabilire una comunicazione DTLS-SRTP su PJSIP

Ora che abbiamo presentato un idea di come implementare il trasporto DTLS-SRTP su PJSIP, è corretto fornire anche una base per stabilire la comunicazione in modo sicuro e semplice. Il ruolo di SIP teoricamente è solo quello di stabilire chi è il server e chi è il client della comunicazione, ma in questo caso sarebbero necessari dei certificati rilasciati da un CA. Se le chiavi utilizzate nell'handshake DTLS sono certificate, i partecipati della comunicazione potranno verificare la reciproca identità consultando il certificate server. Ottenere questo tipo di certificati è complicato e difficile, ma soprattutto è uno sforzo inutile in questo particolare contesto. Seguendo le

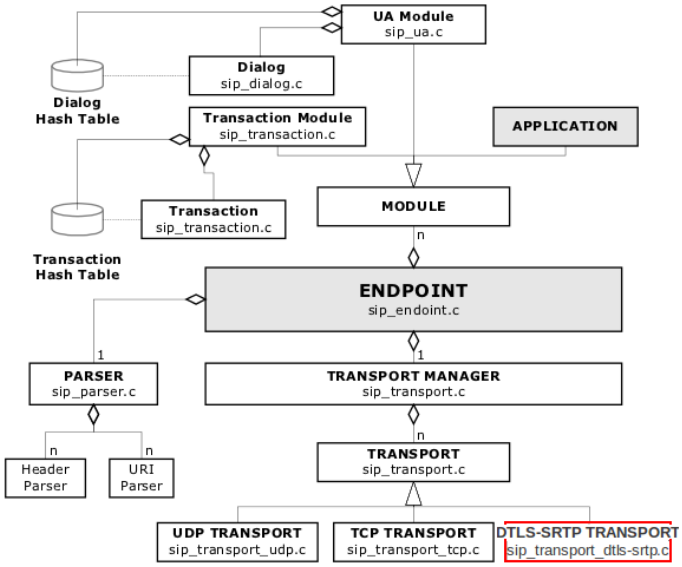


Figura 3.3: DTLS-SRTP su PJSIP

metodologie presenti nel RFC 5763 è possibile evitare agli UA di procurarsi un certificato prodotto da un CA fornendo comunque una sicurezza completa. Questa soluzione si basa sulla possibilità di auto-produrre e auto-firmare i certificati. Considerando quindi che sono gli utenti a generare e firmare i propri certificati è essenziale incorporare un qualche meccanismo che attesti l'identità di tale utente. Il modo più semplice è quello di trasferire l'identificazione su SIP che a questo punto deve garantire un meccanismo di SIP Identity. Lo strumento migliore offerto da SIP è l'Enhanced SIP Identity (RFC 4474); se esso è in uso, l'unico servizio che l'UAC deve supportare nella fase di INVITE è la classica autenticazione HTTP/Digest; Questa funzione è perfettamente supportata da PJSIP, per altro, è presente un framework per gestire tale autenticazione all'interno di pjsipsip_auth.h.

Il proxy, dopo che l'UAC si è autenticato tramite HTTP/Digest aggiunge due nuovi campi al messaggio SIP: Identity-Info: contiene le informazioni

necessarie per acquisire il certificato del proxy. Identity: è il risultato di un algoritmo sha1WithRsaEncryption su una stringa formata da alcuni elementi della richiesta SIP che sono qui di seguito elencati, separati dal carattere $||o\%x7C$ :

l'indirizzo che compare nel campo From;

l'indirizzo che compare nel campo To;

il Call Id contenuto nel campo Call-Id;

il contenuto del campo CSeq facendo attenzione a convertire gli spazi in x20%;

il contenuto del campo Date, anche in questo caso convertendo gli spazi;

il contenuto del campo Contact;

il corpo del messaggio senza apportare modifica.

Di seguito un esempio per comprendere meglio come viene formata la stringa. Ipotizziamo che il messaggio inviato dal UAC al proxy sia il seguente:

```
INVITE sip:bob@biloxi.example.org SIP/2.0
Via: SIP/2.0/TLS pc33.atlanta.example.com;branch=z9hG4bKnashds8
To: Bob <sip:bob@biloxi.example.org>
From: Alice <sip:alice@atlanta.example.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 314159 INVITE
Max-Forwards: 70
Date: Thu, 21 Feb 2002 13:02:03 GMT
Contact: <sip:alice@pc33.atlanta.example.com>
```

```
Content-Type: application/sdp
Content-Length: 147

v=0
o=UserA 2890844526 2890844526 IN IP4 pc33.atlanta.example.com
s=Session SDP
c=IN IP4 pc33.atlanta.example.com
t=0 0
m=audio 49172 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

Di conseguenza la stringa dove poi viene applicato l'algoritmo è:

```
sip:alice@atlanta.example.com|sip:bob@biloxi.example.org|
a84b4c76e66710|314159 INVITE|Thu, 21 Feb 2002 13:02:03 GMT|
sip:alice@pc33.atlanta.example.com|v=0
o=UserA 2890844526 2890844526 IN IP4 pc33.atlanta.example.com
s=Session SDP
c=IN IP4 pc33.atlanta.example.com
t=0 0
m=audio 49172 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

Presumendo che il risultato dell'algoritmo su tale stringa sia:

```
ZYNBbHC00VMZr2kZt6VmCvPonWJMGvQTBDqghoWeLxJfzB2a1pxAr3VgrB0SsSAa
  ifsRdiOPoQZYOy2wrVghuhcsMbHWUSFxI6p6q5TOQXHMmz6uEo3svJsSH49thyGn
  FVcnyaZ++yRlBYYQTLqWzJ+KVhPKbfU/pryhVn9Yc6U=
```

il proxy inviera al UAS un messaggio SIP cosi formato:

```
INVITE sip:bob@biloxi.example.org SIP/2.0
Via: SIP/2.0/TLS pc33.atlanta.example.com;branch=z9hG4bKnashds8
To: Bob <sip:bob@biloxi.example.org>
From: Alice <sip:alice@atlanta.example.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 314159 INVITE
Max-Forwards: 70
Date: Thu, 21 Feb 2002 13:02:03 GMT
Contact: <sip:alice@pc33.atlanta.example.com>
Identity:
  "ZYNBbHC00VMZr2kZt6VmCvPonWJMGvQTBDqghoWeLxJfzB2a1pxAr3VgrB0SsSAa
  ifsRdiOPoQZYOy2wrVghuhcsMbHWUSFxI6p6q5TOQXHMmz6uEo3svJsSH49thyGn
  FVcnyaZ++yRlBYYQTLqWzJ+KVhPKbfU/pryhVn9Yc6U="
Identity-Info: <https://atlanta.example.com/atlanta.cer>;alg=rsa-sha1
Content-Type: application/sdp
Content-Length: 147

v=0
o=UserA 2890844526 2890844526 IN IP4 pc33.atlanta.example.com
s=Session SDP
c=IN IP4 pc33.atlanta.example.com
t=0 0
m=audio 49172 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

dove https://atlanta.example.com/atlanta.cer è l'indirizzo del certifi-

cato del proxy e rsa-sha1 è l'algoritmo usato dal proxy. L'UAS che riceve questo messaggio deve essere in grado di verificare l'identità del chiamante. Per far questo deve calcolare la stringa con lo stesso metodo spiegato sopra e computarne il digest con l'algoritmo indicato nel campo identity info, decifrare il campo identity con la chiave pubblica del proxy e infine confrontare il digest ottenuto dalla firma con quello ottenuto dalla stringa, se tra i due non ci sono differenze l'identità del chiamante è confermata. La soluzione anche in questo caso è affidata all'uso di OpenSSL, ma questo non dovrebbe più essere un problema considerando che esso è un requisito fondamentale per fornire DTLS-SRTP.

Prima ancora di effettuare queste operazioni un UA deve essere in possesso del certificato auto-firmato. Le funzioni OpenSSL[24] da invocare a questo scopo sono brevemente spiegate di seguito:

- EVP_PKEY_new: si occupa di allocare spazio per una struttura di tipo EVP_PKEY dove verranno memorizzate le chiavi.
- X509_new: alloca spazio per memorizzare la struttura che conterrà il certificato X509
- RSA_generate_key genera una coppia di chiavi e le restituisce all'interno di una struttura di tipo RSA
- EVP_PKEY_assign_RSA: si occupa di prelevare le chiavi dalla struttura restituita dalla funzione RSA_generate_key e di memorizzarla in una struttura di tipo EVP_PKEY.
- X509_set_pubkey, X509_set_version, X509_NAME_add_entry_by_txt: servono per manipolare il certificato. La prima serve per settare

a chiave pubblica ottenuta da RSA_generate_key, la seconda setta la versione del certificato, mentre l'ultima si occupa di aggiungere le entrate tipiche di un certificato.

La firma infine si ottiene invocando semplicemente il metodo X509_sign che prende in input il certificato, la coppia di chiavi e un puntatore a EVP_MD (una struttura che contiene i dettagli dell'algoritmo di hash utilizzato). Il contenuto dell'attributo subjectAltName all'interno del certificato può utilizzare l'URI dell'utente, ma questo non è obbligatorio, in quanto l'integrità del certificato è garantita dal suo fingerprint che viene inviato nel SDP.

Il fingerprint del certificato (nell'esempio corrispondente a 4A:AD:B9:B1-:3F:82:18:3B:54:02:12:DF:3E:5D:49:6B:19:E5:7C:AB) va allegato all'SDP del messaggio SIP dove andranno inseriti anche i campi necessari a descrivere la sessione da instaurare come da esempio sotto riportato sotto.

```
V=0
o=- 1181923068 1181923196 IN IP4 ua1.example.com
s=example1
c=IN IP4 ua1.example.com
a=setup:actpass
a=fingerprint: SHA-1 \4A:AD:B9:B1:3F:82:18:3B:54:02:12:DF
      :3E:5D:49:6B:19:E5:7C:AB
t=0 0
m=audio 6056 RTP/AVP 0
a=sendrecv
a=tcap:1 UDP/TLS/RTP/SAVP RTP/AVP
```

```
a=pcfg:1 t=1
```

Ora rimane da decidere come dividere il lavoro e dove implementarlo.

Una possibilità è quella di implementare la creazione delle chiavi, e del certificato all'interno dell'applicazione PJSIP e delegare invece l'inserimento dei campi necessari nel SDP, la verifica del fingerprint e dell'identità di un UA ad un modulo separato che avrà priorità di livello applicazione (PJSIP MOD PRIORITY APPLICATION) e verrà creato subito dopo l'operazione di REGISTER. Questo modulo intercetta ogni messaggio SIP diretto verso il proxy o ricevuto da quest'ultimo e si occupa a seconda dei casi di aggiungere il fingerprint o verificare l'identità e certificato. Esso deve essere in grado di comunicare con il trasporto DTLS-SRTP per avere le informazioni riguardo i dettagli del DTLS handshake.

Il trasporto da utilizzare per la sessione non deve essere per forza TLS in quanto l'integrità è garantita su tutto il corpo del messaggio dalla firma apposta dal proxy server, ma rimane comunque il trasporto da preferire per evitare un attacco MiTM tra UAC e proxy.

### 3.2.3 ZRTP SU PJSIP

Un altro modo di scambiare le chiavi che poi saranno utilizzate per una connessione SRTP è ZRTP. Questa soluzione è molto più facile da offrire su PJSIP rispetto alla precedente in quanto ci sono già delle implementazioni di ZRTP che hanno deciso di adattarsi per funzionare con PJSIP. Le due implementazioni in questione sono GNU ZRTP e Zorg [25]. Entrambe forniscono una patch che modifica PJSIP dove necessario, e spiegano bene i passi da seguire per applicarle. Zorg è più

adatta ad applicazioni mobili, infatti può girare perfettamente anche su sistemi operativi Symbian S60.

## 3.3 DTLS su SIP

Datagram Transport Layer Securety (DTLS) non è standardizzato in nessun RFC come trasporto per le sessioni SIP, ma teoricamente è possibile fornire supporto per tale trasporto al di sopra di PJSIP. Infatti SIP è un protocollo di applicazione che può essere eseguito sia su un trasporto di tipo stream che uno di tipo datagram, compresi UDP e TCP, e come abbiamo visto per l'implementazione di DTLS-SRTP è possibile aggiungerlo a PJSIP imitando il trasporto TLS. Prima di far questo però bisognerebbe estendere il protocollo SIP aggiungendo nuovi campi di intestazioni adatti a contrattare tale tipo di trasporto. Il campo Via nell'header SIP identifica il tipo di trasporto. Quindi per prima cosa è necessario estendere l'RFC 3261 per definire un nuovo valore, che può essere "DTLS-UDP", ottenendo cosi un campo Via: SIP/2.0/DTLS-UDP example.com:5060

# Conclusioni

Dopo una breve descrizione del protocollo SIP nel primo capitolo, ho affrontato la questione sicurezza nel caso specifico di una comunicazione SIP, individuando sia i meccanismi per proteggere lo strato di sessione che quelli per proteggere lo strato di trasporto. Da questo sono emerse alcune problematiche, come ad esempio la fragilità della sicurezza su base hop-by-hop, l'overhead causato da una possibile implementazione del trasporto DTLS, e il problema della distribuzione dei certificati.

Per ognuna di queste problematiche ho cercato una soluzione tra quelle disponibili; ciò ha evidenziato in particolar modo la validità di ZRTP e DTLS-SRTP se affiancato ad un meccanismo per gestire la validità dei certificati.

Entrambi sono due protocolli specifici per lo scambio chiavi necessario a stabilire una connessione SRTP, con delle peculiarità molto simili. Infatti a differenza di altre soluzioni come ad esempio SRTP/SDES, lo scambio avviene in maniera diretta tra i peer partecipanti alla comunicazione e non attraverso il protocollo di segnalazione. Questo permette di evitare qualsiasi tipo di intromissione da parte di terzi.

A seguito di questo capitolo è stata presa in esame la fattibilità di un implementazione sullo stack PJSP di alcuni dei meccanismi presi in

esame.

Ciò ha messo in luce l'ottima estensibilità di PJSIP che è in grado di adattarsi perfettamente con implementazioni del protocollo ZRTP già esistenti attraverso l'ausilio di semplici patch che tra l'altro sono già messe a disposizione dalle implementazioni stesse.

Anche nel caso di DTLS-SRTP per cui non esiste ancora alcuna implementazione completa e portabile su diversi sistemi, PJSIP riesce ad adattarsi perfettamente, riuscendo per di più ad implementare quanto riportato nel RFC 5763 per poter aggirare il problema della distribuzione dei certificati.

Detto questo posso concludere che PJSIP è perfettamente in grado di estendersi e adattarsi per supportare i migliori strumenti di sicurezza per SIP sfruttando librerie esterne, quindi rimane solo una scelta dell'utente decidere a quali appellarsi.

Questa mia analisi potrebbe essere ampliata in futuro studiando la fattibilità di un implementazione di un meccanismo per la gestione dei certificati come ad esempio il recente RFC 6072, che potrebbe andare a sostituire l'RFC 5763 studiato in questa tesi per aggirare il problema della distribuzione dei certificati. L'RFC 6072 renderebbe possibile la sincronizzazione delle chiavi private su diversi dispositivi tramite l'uso di SUSCRIBE e NOTIFY.

# Bibliografia

[1] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. "SIP: Session Initiation Protocol", RFC 3261, June 2002.

[2] M. Handley, V. Jacobson, and C. Perkins. "SDP: Session Description Protocol", RFC 4566, July 2006.

[3] S. Donovan. "The SIP INFO Method", RFC 2976, October 2000.

[4] J. Rosenberg, and H. Schulzrinne. "Reliability of Provisional Responses in Session Initiation Protocol (SIP)", RFC 3262, June 2002.

[5] A. Roach. "Session Initiation Protocol (SIP)-Specific Event Notification", RFC 3265, June 2002.

[6] J. Rosenberg, "The Session Initiation Protocol (SIP) UPDATE Method", RFC 3311, September 2002.

[7] B. Campbell, J. Rosenberg, H. Schulzrinne, C. Huitema, D. Gurle. "Session Initiation Protocol (SIP) Extension for Instant Messaging", RFC 3428, December 2002.

[8] R. Sparks. “The Session Initiation Protocol (SIP) Refer Method”, RFC 3515, April 2003.

[9] J. Arkko, V. Torvinen, G. Camarillo, A. Niemi, T. Haukka “Security Mechanism Agreement for the Session Initiation Protocol (SIP)”, RFC 3329, January 2003. January 2003

[10] E. Rescorla, N. Modadugu. “ Datagram Transport Layer Security”, RFC 4347, April 2006

[11] R. Oppliger. “SSL and TLS Theory and Practice”, August 2009

[12] J. Franks, P. Hallam-Baker, J. Hostetler, S. Lawrence, P. Leach, A. Luotonen, L. Stewart. “HTTP Authentication: Basic and Digest Access Authentication”, RFC 2617, June 1999.

[13] J. Peterson, and C. Jennings. “Enhancements for Authenticated Identity Management in the Session Initiation Protocol (SIP)”, RFC 4474, August 2006.

[14] B. Ramsdell. “Secure/Multipurpose Internet Mail Extensions (S/MIME) Message Specification” ,RFC 3851, July 2004

[15] C. Jennings, J. Fischl. “Certificate Management Service for the Session Initiation Protocol (SIP)”, RFC 6072, February 2011

[16] J. Lennox. “Connection-Oriented Media Transport over the Transport Layer Security (TLS) Protocol in the Session Description Protocol (SDP)”, RFC 4572, July 2006.

[17] H. Schulzrinnements, S. Casner, R. Frederick, V. Jacobson. “RTP: A Transport Protocol for Real-Time Applications”, RFC 3550 July 2003.

[18] M. Baugher. "The Secure Real-time Transport Protocol (SRTP)", RFC 3711, March 2004.

[19] F. Andreasen, M. Baugher, and D. Wing. "Session Description Protocol (SDP) Security Descriptions for Media Streams", RFC 4568, July 2006.

[20] D. McGrew, E. Rescorla."Datagram Transport Layer Security (DTLS) Extension to Establish Keys for the Secure Real-time Transport Protocol (SRTP)", RFC 5764, May 2010.

[21] J. Fischl, H. Tschofenig, E. Rescorla. "Framework for Establishing a Secure Real-time Transport Protocol (SRTP) Security Context Using Datagram Transport Layer Security (DTLS)", RFC 5763, May 2010.

[22] P. Zimmermann, A. Johnston, J. Callas. "ZRTP: Media Path Key Agreement for Unicast Secure RTP", RFC 6189, April 2011.

[23] B. Prijiono. "PJSIP Developer's Guide", March 2006.

[24] OpenSSL Documents, "http://www.openssl.org/docs/".

[25] Zorg implementation of the ZRTP protocol. "http://www.zrtp.org/".

[26] Alan B. Johnston. "SIP: Understanding the Session Initiation Protocol Third Edition", 2009.